2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Automation of Reusable Protocol-Agnostic Performance Analysis in UVM Environments

Daniel Carrington, Alan Pippin, Timothy Pertuit

Hewlett Packard Enterprise

accellera SYSTEMS INITIATIVE

Outline

Performance measurement is the key to performance verification

Our solution

Automated Performance Testing (Autoperf)

- Performance Verification Workflow Improvements
- Performance Measurement Tools (Bandwidth, Utilization, Latency)

Performance Verification

- Develop initial design and testbench
- Run performance tests once your design is “pretty functional”
- Measure the DUT performance realized in simulation
- Iterate until everything is fixed or you run out of time

Performance Verification

- Develop initial design and testbench
  - Solved by UVM
- Run performance tests once your design is “pretty functional”
  - Partly solved by UVM (with help expected from Portable Stimulus)
- Measure the DUT performance realized in simulation
  - Not solved; write a script yourself or work within the limits of industry tools
- Iterate until everything is fixed or you run out of time
  - Coverage tooling, checklists, management

Performance Verification

State of bandwidth measurement

- Protocol-specific commercial tools
- Easy enough to read wave database and write a script

State of latency measurement

- Round-trip latency measurement: match requests to responses
- More granular measurements: do it manually, or write limited latency tests

Our Solution: Autoperf

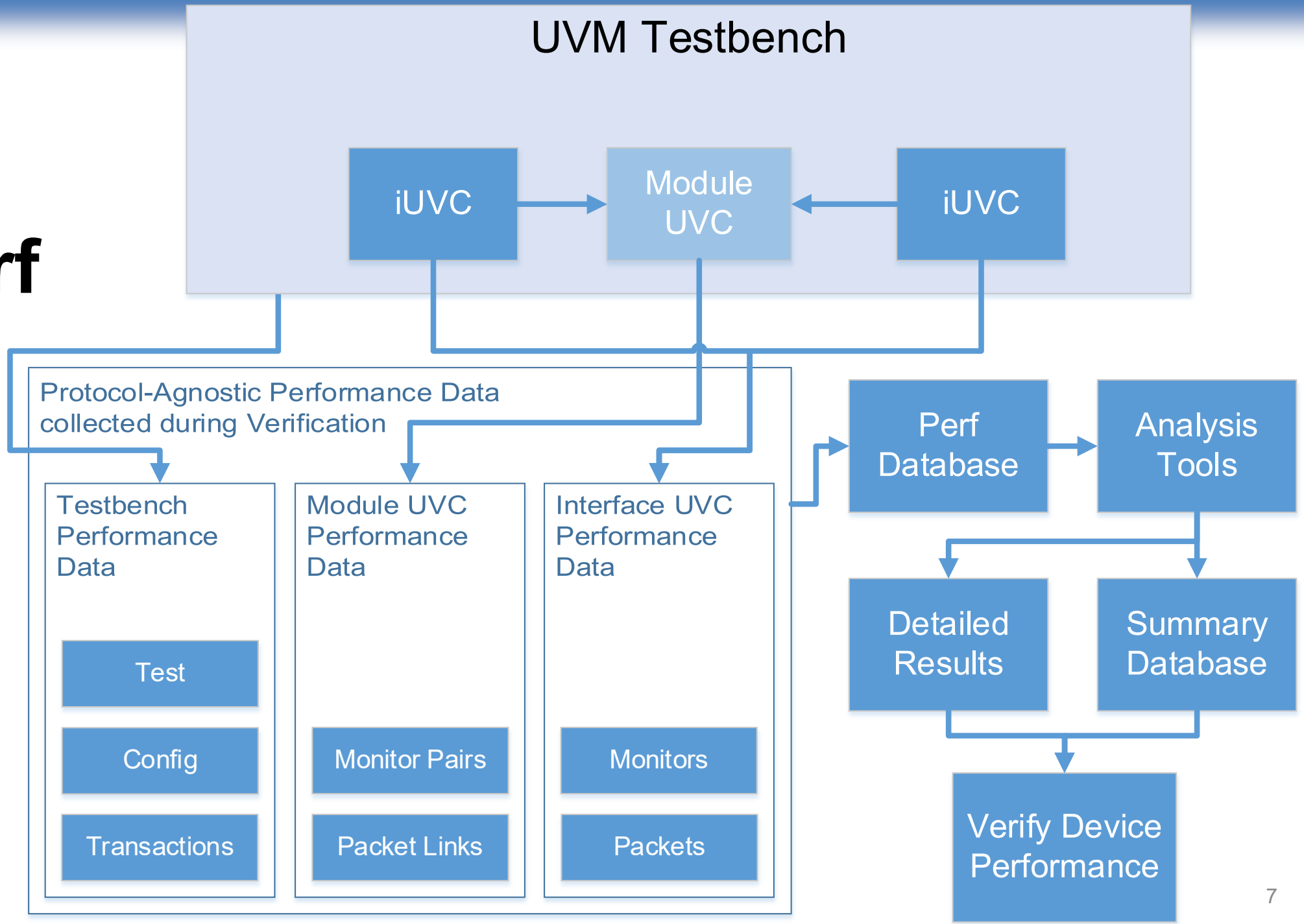
Value from UVM reuse

- Define a performance measurement API from UVM components
- Leverage UVM configuration
- Enable reuse of performance measurement

Set and track performance goals

- Architectural metrics and measurements across multiple tests
- Optional pass/fail criteria on regression tests
- Logs performance data into database for analysis

Autoperf


UVM Testbench
iUVC
Module UVC
iUVC
Protocol-Agnostic Performance Data collected during Verification
Testbench Performance Data
Test
Config
Transactions
Module UVC Performance Data
Monitor Pairs
Packet Links
Interface UVC Performance Data
Monitors
Packets
Perf Database
Analysis Tools
Detailed Results
Summary Database
Verify Device Performance

Autoperf

- Performance instrumentation implemented by UVM testbench
  - Doing performance instrumentation once per iUVC is a fantastic model
  - Doing latency measurement in each module UVC at block-level and composing those into a system simulation Just Works and it's great
- Performance measurement tool is protocol-agnostic
  - Measurement tool is easy to understand
  - Validating performance model is easy

Components of Autoperf

Verification Libraries

- Specman using UVM-e
- SystemVerilog using UVM-ML and UVM-SV
- 3.5kloc

API

- Structured log messages report data from sim
- Performance database schemas

Scripts

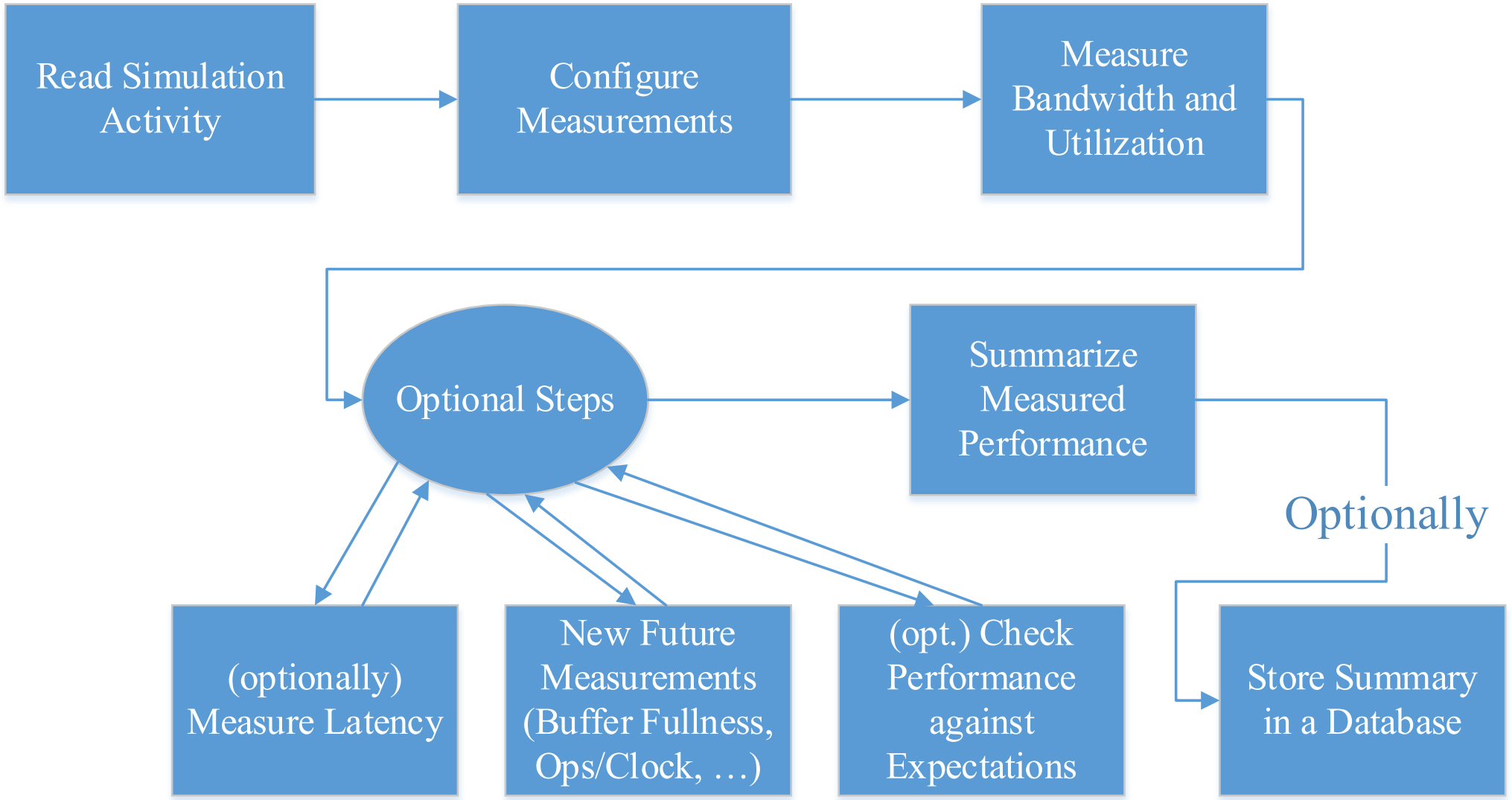
- Process performance data (13.5kloc Perl, SQL)
- Web database interface (3.5kloc Perl, JS, HTML)
- Autoperf unit tests (3kloc Perl)

Autoperf Verification API

- UVCs use Autoperf classes to print messages to an Autoperf file
- Messages are processed and performance results are produced

UVM Component	Autoperf Log Message	Printed During	Purpose
Testbench	test_message	Time 0	Provide user-defined parameters describing the test
Testbench	config_message	Time 0	Override any special Autoperf config options for this test
uvm_monitor	mon_message	First clock cycle	Declare existence of hardware interface
uvm_monitor	pkt_message	Main test	Report packet activity
uvm_scoreboard	mon_mon_pair_msg	First clock cycle	Declare relation of two hardware interfaces
uvm_scoreboard	pkt_pkt_link_msg	Main test	Report relation of two previous pkt_messages
	trans_message	Main test	Identify packet as originating from a sequence

Autoperf Post-Simulation Flow



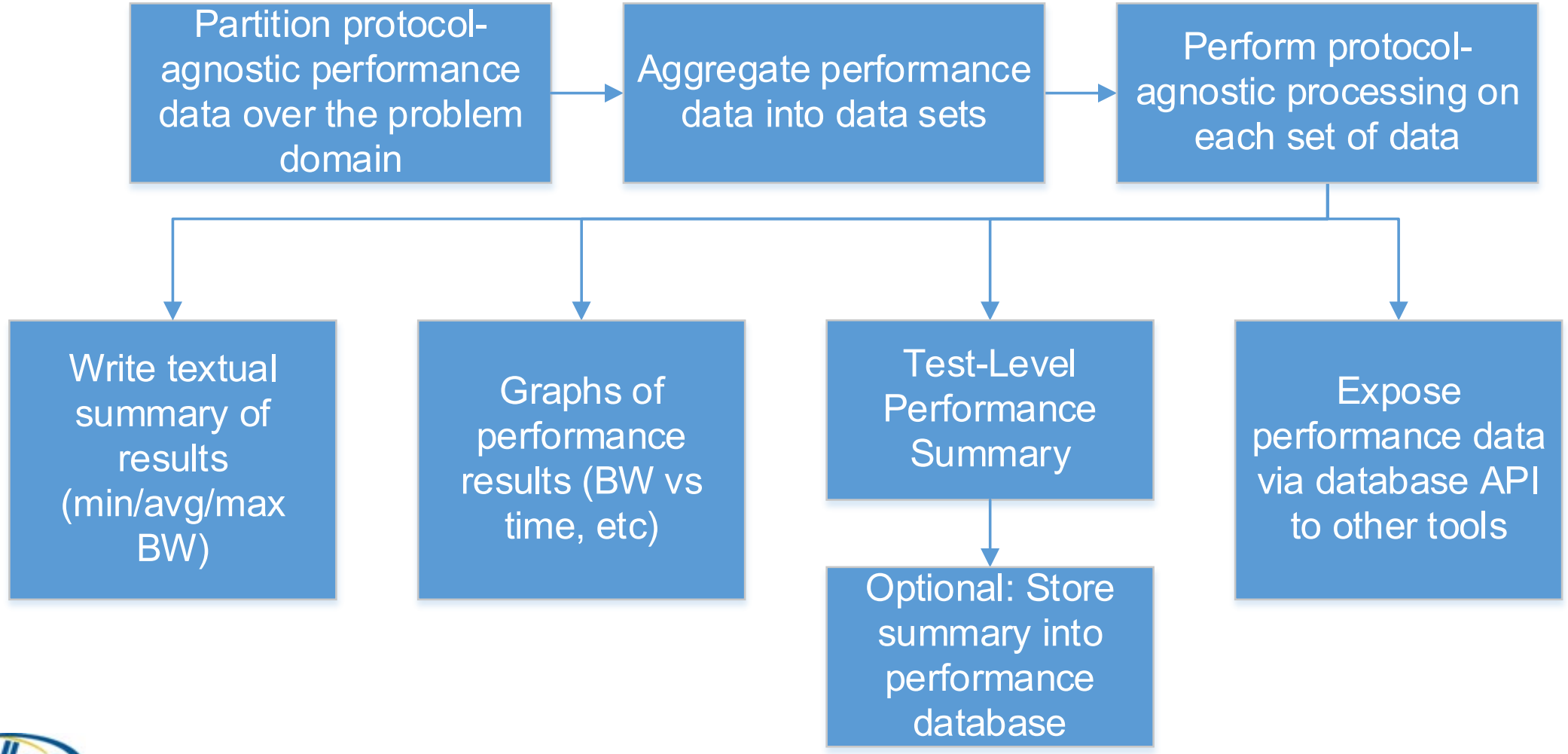
“Just Measure Performance”

- Reusable performance measurement in verification components
  - Expose performance data from the simulation environment
  - Where to measure? What activity occurred? How much? When?

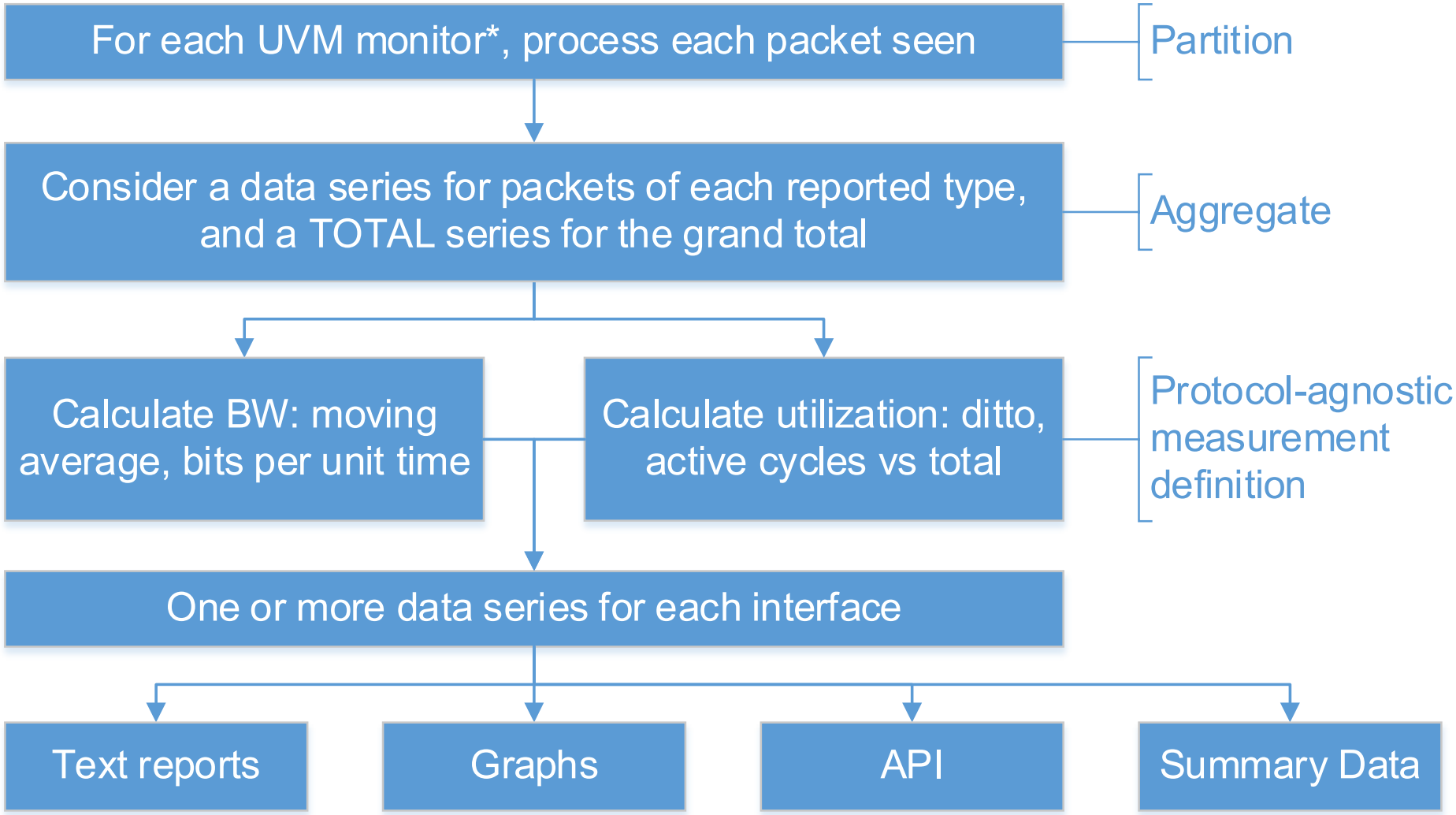
- Reusable Bandwidth, Utilization, and Latency analysis tools
  - Consume performance data without configuration or special set-up

- Accessible by any user of UVM
  - Purchase your VIP from anyone or develop your own

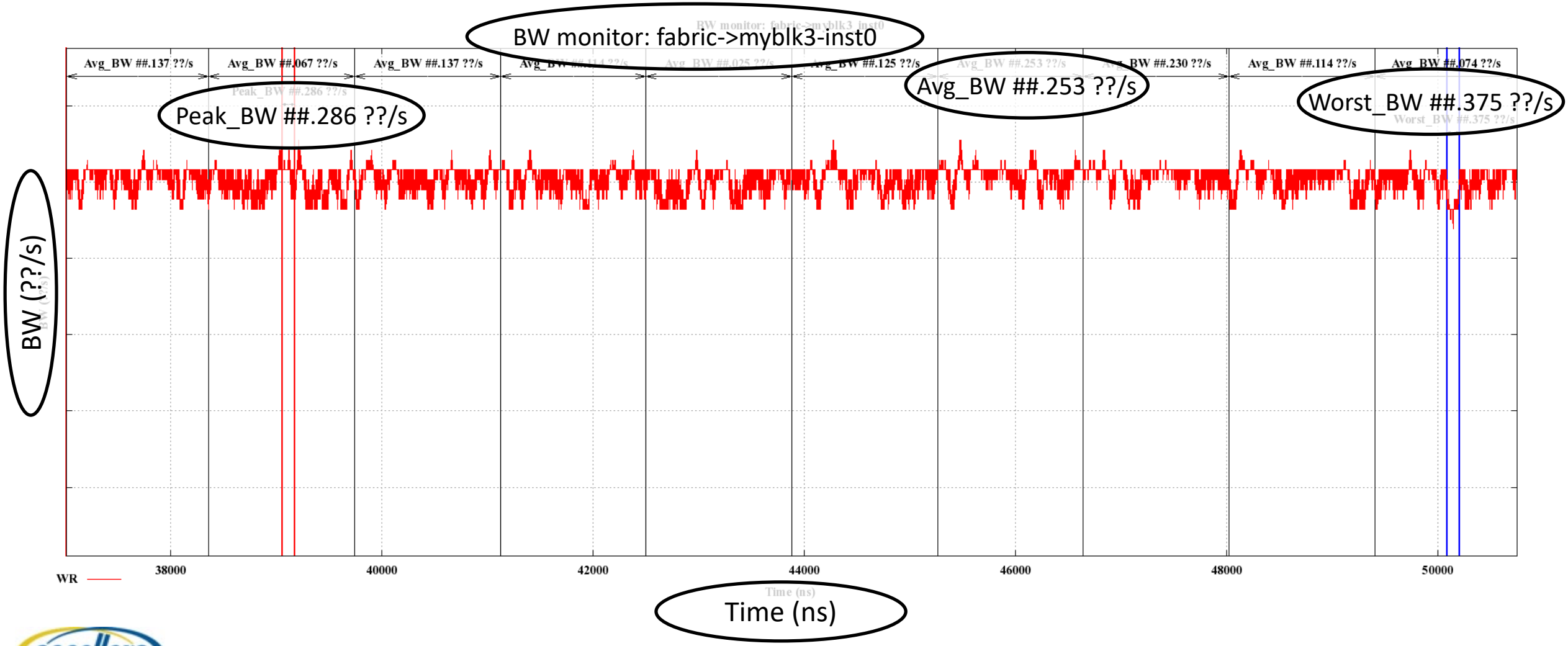
Protocol-Agnostic Performance Measurement



Protocol-Agnostic Performance Measurement

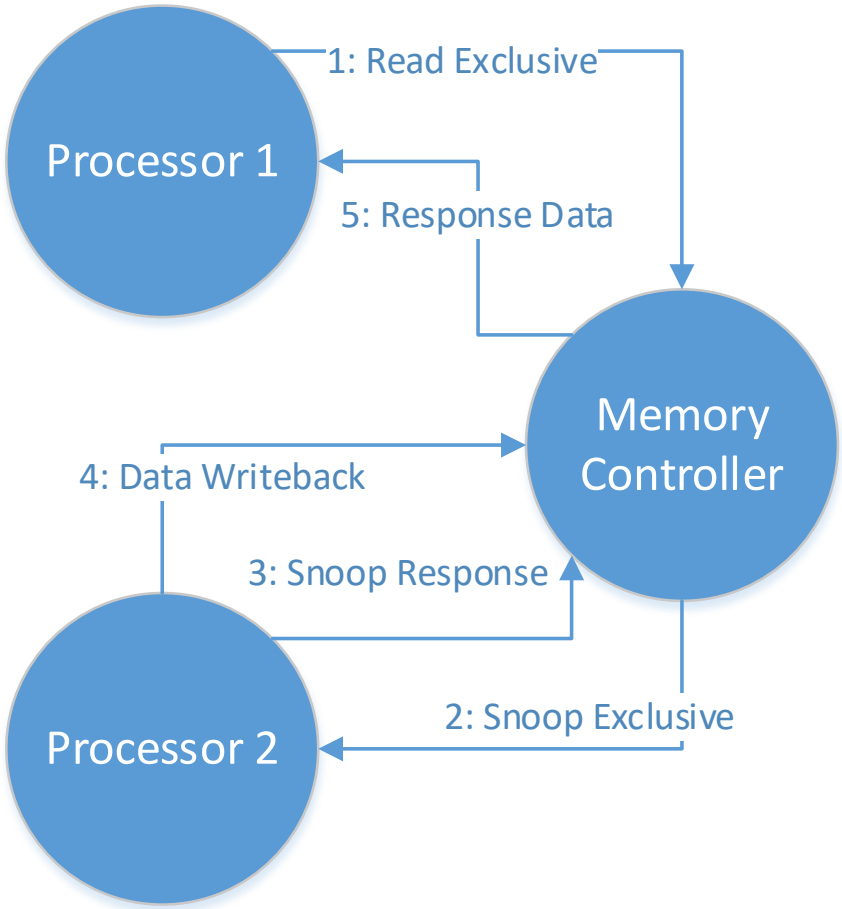


Autoperf: Bandwidth and Utilization Results



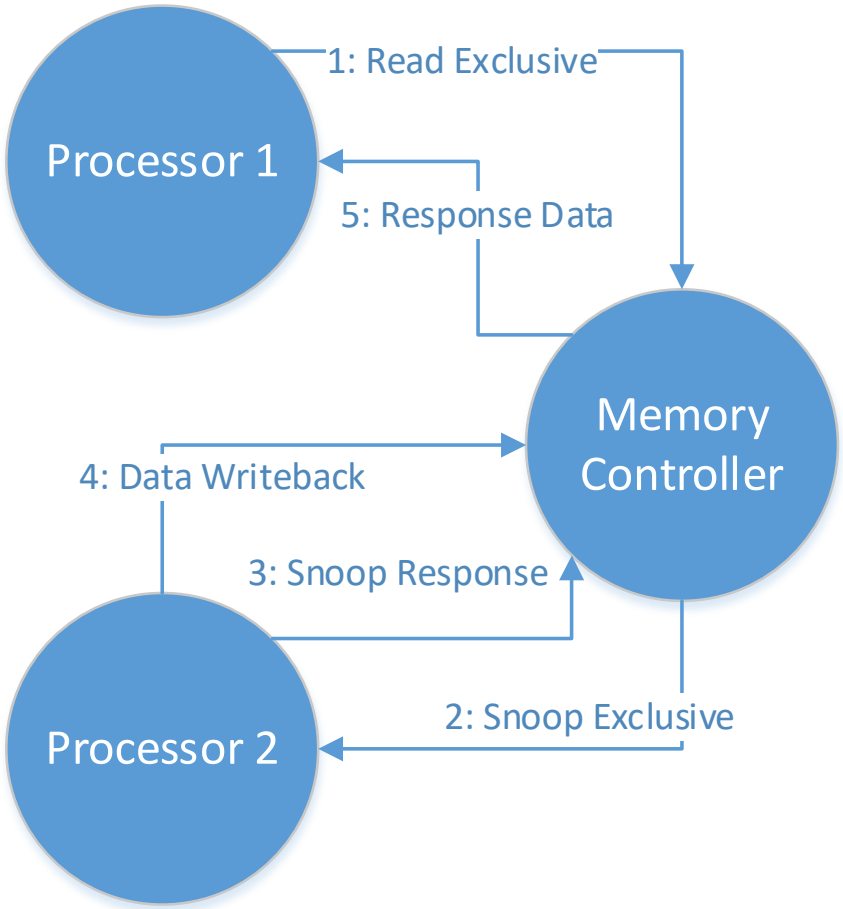
Autoperf Latency Measurement

- Cache coherency is everyone's favorite tricky performance verification task
- Measuring latency and figuring out latency defects can be challenging
- Read+Snoop+Rsp has a lot of places to introduce latency

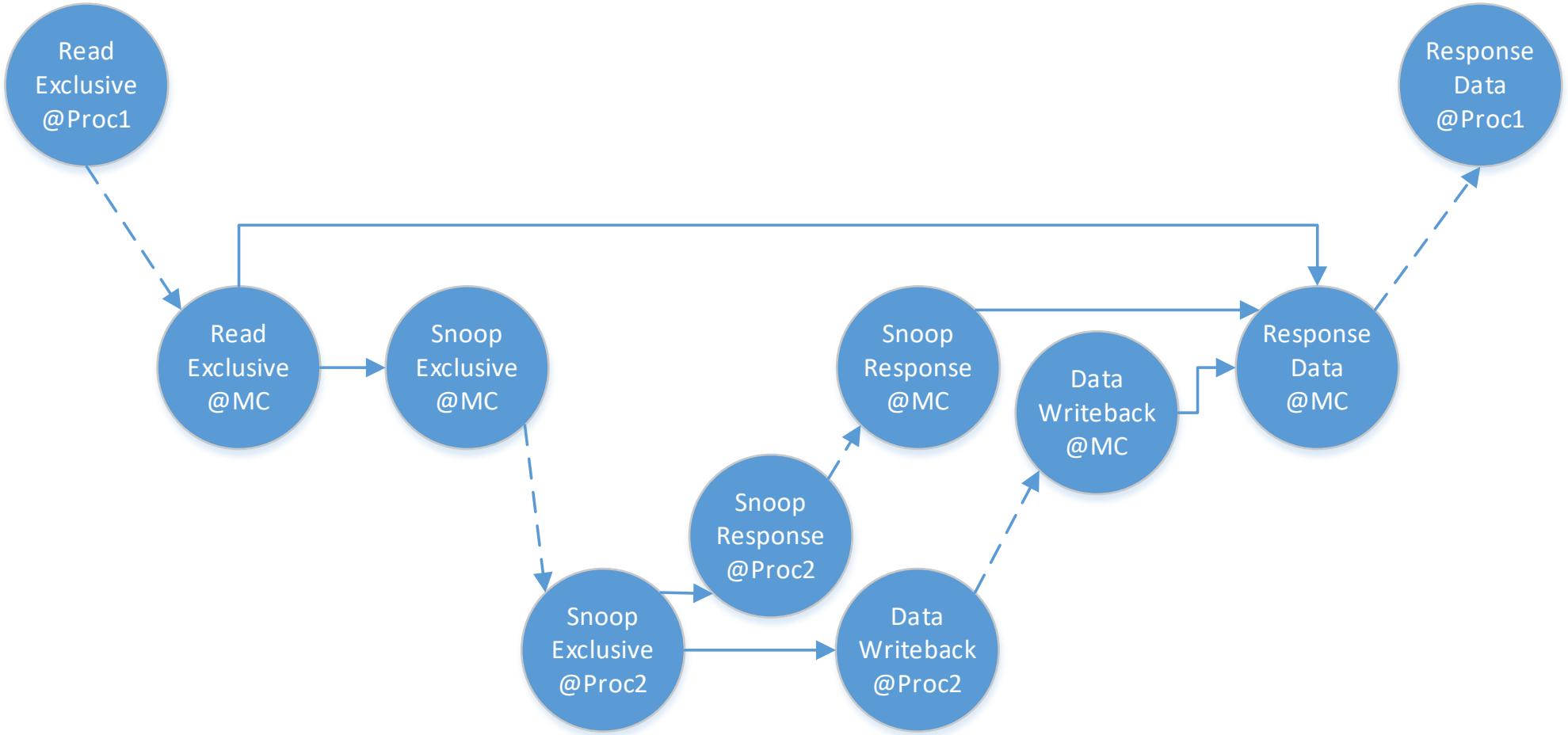


Autoperf Latency Measurement

- Big Questions:
  - How long did each phase take?
  - Which phase caused the performance issue?
  - What logic is causing the problem?

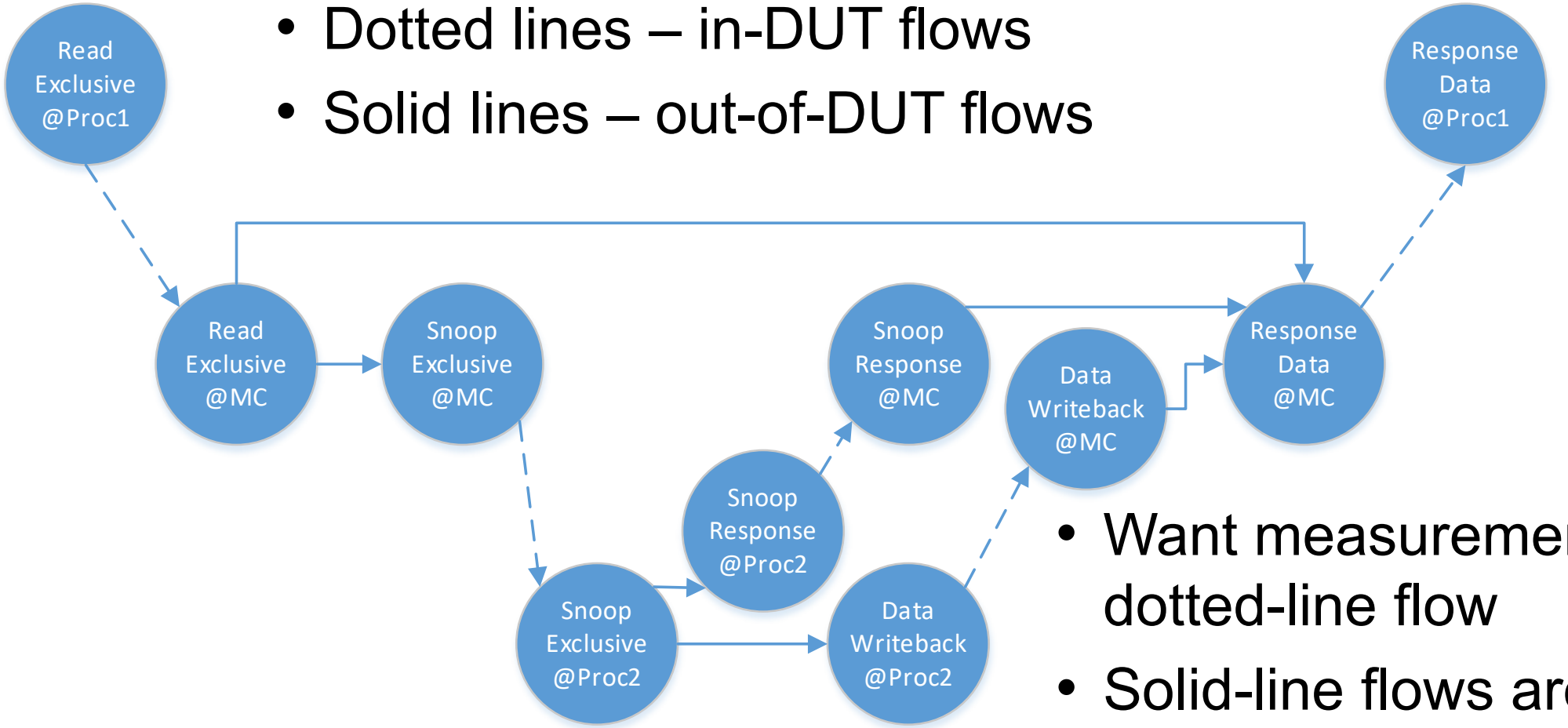


Autoperf Latency Measurement



Autoperf Latency Measurement

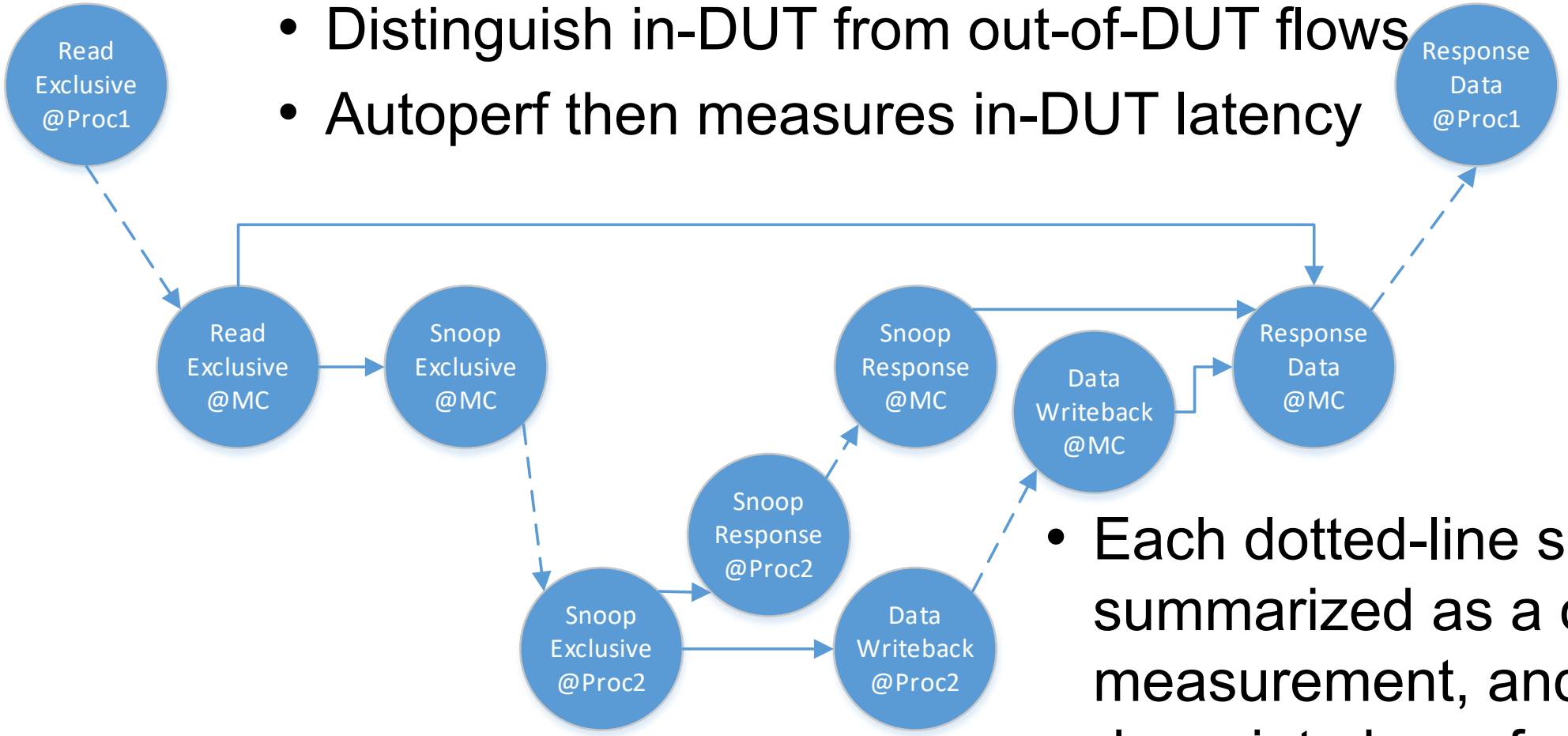
- Dotted lines – in-DUT flows
- Solid lines – out-of-DUT flows



- Want measurements of each dotted-line flow
- Solid-line flows are someone else's problem at this stage

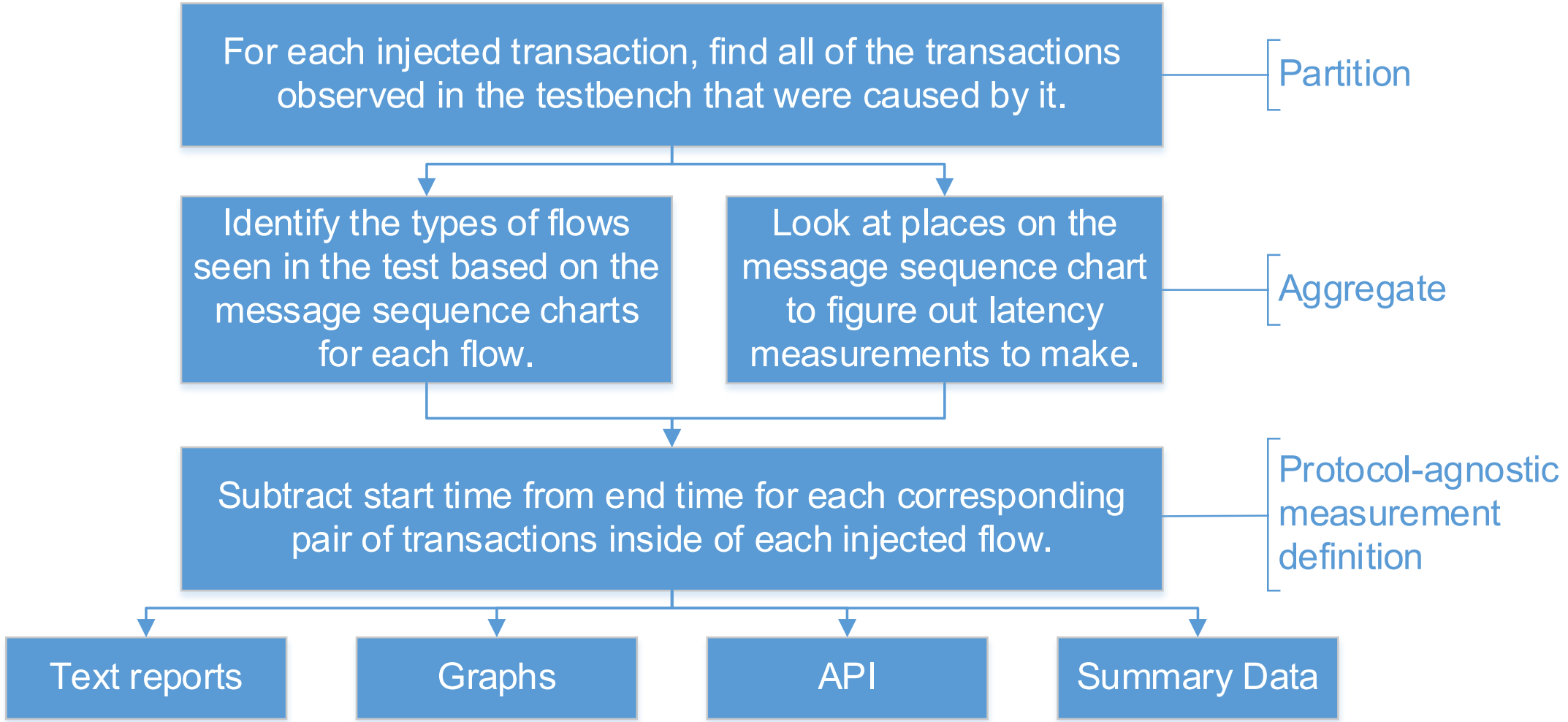
Autoperf Latency Measurement

- Distinguish in-DUT from out-of-DUT flows
- Autoperf then measures in-DUT latency

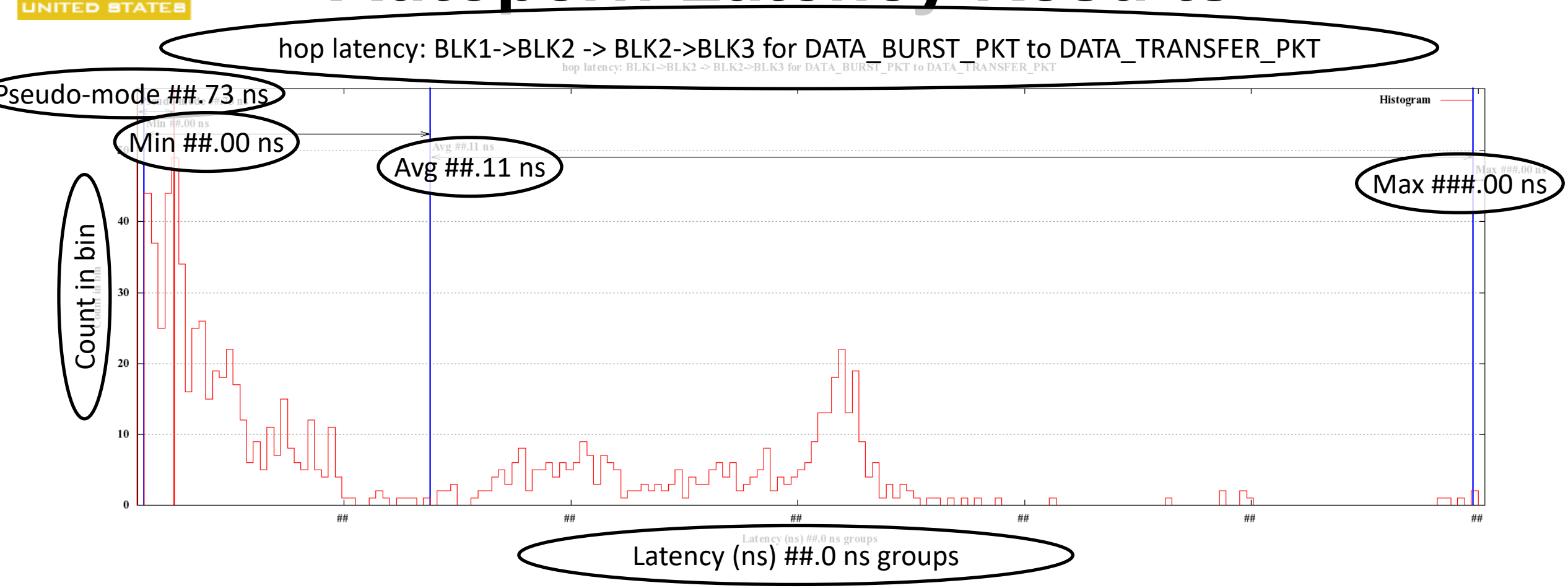


- Each dotted-line subflow is summarized as a one-way measurement, and broken down into hops for bug RCA

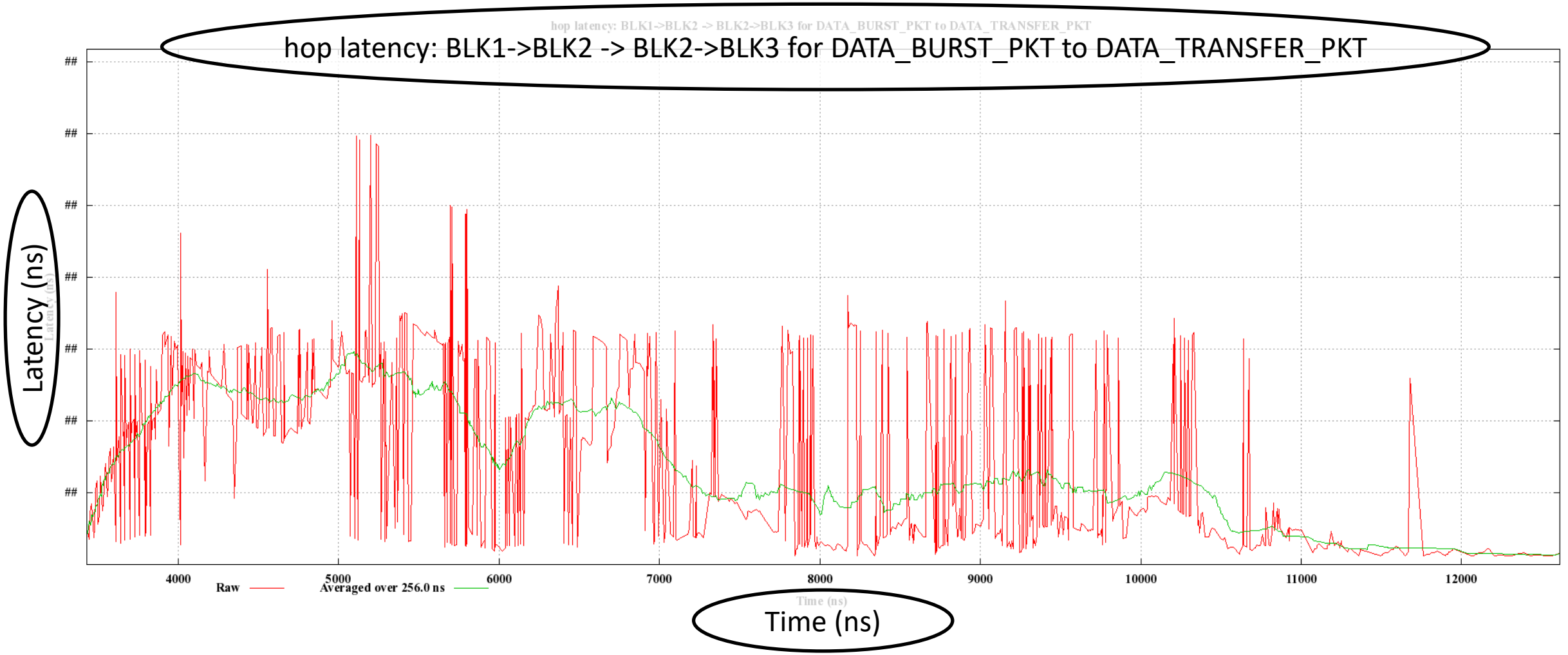
Protocol-Agnostic Performance Measurement



Autoperf: Latency Results



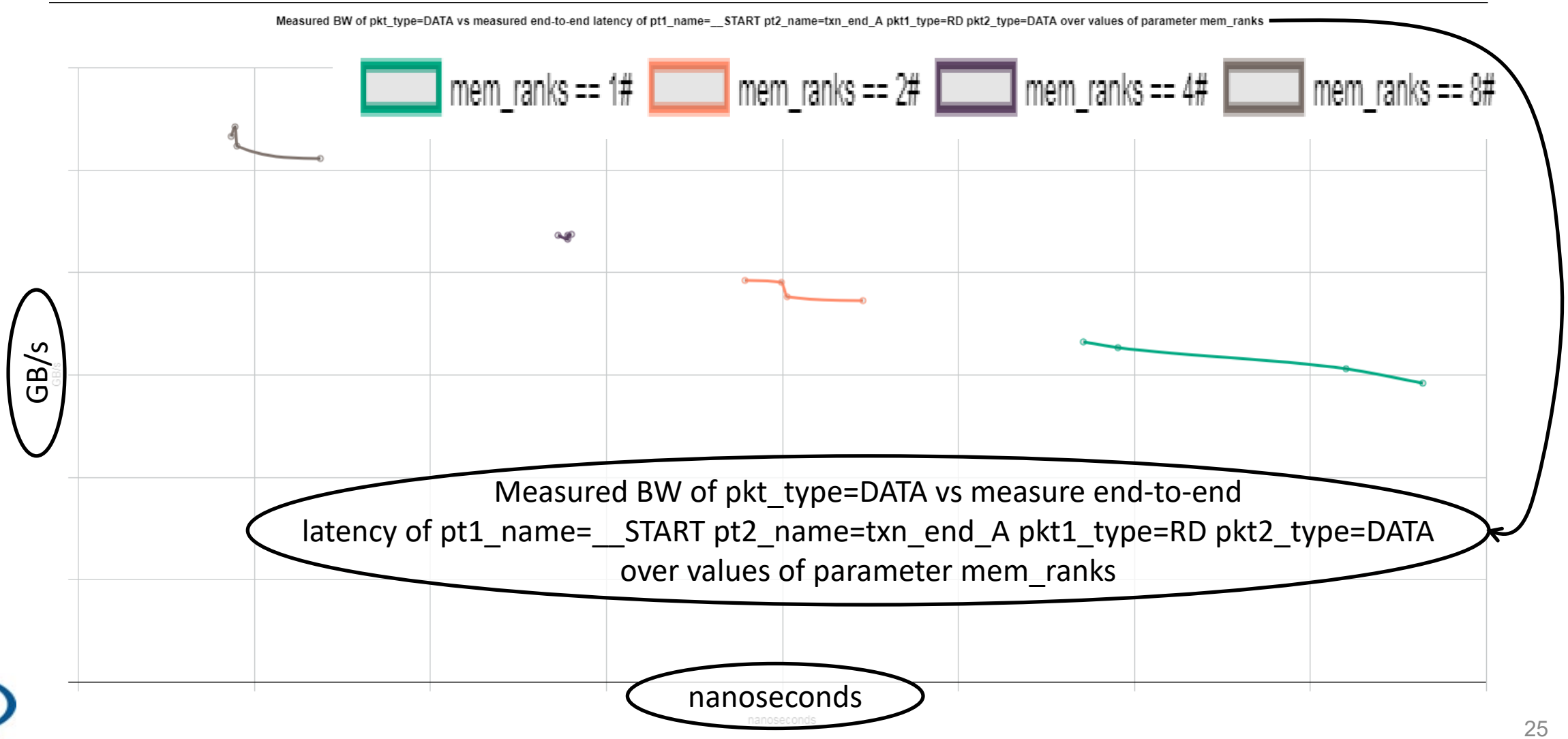
Autoperf: Latency Results



Reusability Limitations

- Interface UVCs need to summarize the wire format in terms of “application data bits” and “overhead bits”
  - Software-defined transfer formats may not be tractable to measure
- Variable clock speeds not explicitly supported
- Assumes that each wire transfers one bit per clock period
    - Easy to work around for DDR interfaces
- Difficulty of doing latency modeling in the Module UVC can vary depending on the application and your scoreboard implementation

Our Solution: Results



Autoperf

- Minimum investment enables bandwidth/utilization measurements
  - Data mover block can measure output bandwidth vs. control utilization
  - Serial link block can measure link utilization achieved

- Optional pass/fail criteria for individual tests
  - Some tests are more important than others
  - Latency regressions are very important to us
  - Bandwidth measurements in directed random testing can be noisy

Autoperf

- Finer data granularity available than round-trip latency measurements
  - Round-trip: Time from sending read request to receiving read response
  - One-way: Time from sending read request until it reaches destination
  - Hop: Time it takes for read request to traverse each block of my design

- Ability to analyze the root cause of bandwidth issues using latency data
  - Can find head-of-line blocking issues by looking for the hop latency with a suspicious distribution of values
  - Can reduce time to fix performance bugs significantly

Future Work

- Power Optimization detection and awareness
  - Lane width reduction
  - Dynamic frequency scaling
    - Should the database schema/log format implement clocking domains?
  - Fine-grained clock gating
    - Should this be scored as “partially utilized”?
    - Should this be scored as “utilized by a CLOCK_DISABLED packet”?
    - Should this be its own separate measurement? Correlation with utilization?
- Modeling, modeling, modeling

Future Work

- Instructions Per Clock measurement
  - Perhaps this is easily implemented as a (verification-only) interface which can retire instructions at a certain maximum rate?
    - Report the utilization of this interface
  - Producing a measurement that is ***clear*** is of greater importance!
  - Formalizing our performance models is also important!

- Support for CPU design was not an initial design goal for Autoperf

Future Work

- Buffer fullness measurement capability
  - Count transactions entering and leaving a block
  - Graph on-chip RAM and register file utilization over time
  - Problem: support different kinds of buffer allocation strategies
  - Problem: pipelined network-on-chip
    - How to account for multiple storage devices inside of a logical “block”?
    - Combinatorial explosion of “allocation” and “residency” strategies
  - Problem: sequence number acknowledgements
  - Is this an application that we need at HPE to verify performance?

Future Work

- Other modeling improvements:
- Simplify latency measurement setup workflow
  - Infer packet causation from other properties
  - Updating scoreboards is highly reusable but can be difficult
- Improve monitoring of performance in test
  - Defining test failure criteria seems insufficient
  - Spreadsheet dashboard defining your test plan and measurements?
  - Consider the data entry burdens for any use of this kind of feature

Future Work

- Should this be a complement to the UVM standard?
  - Pro: reusable performance goodness
  - Pro: adds value to commercial VIP offerings
  - Con: I would not recommend SV or DPI implementation of measurement
    - We have had success with post-processing simulation results into Autoperf format and getting performance results out of that
  - Con: could be difficult to update the performance model
    - EG remove clock_period_ps field from monitor log message, add clock domains to implement support for dynamic frequency adjustment
  - Pragmatically: disrupts EDA industry investment in performance tools